IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:				
	Yves Gattegno, et al.	§	Confirmation No.	6443
		§		
Serial No.:	10/593,262	§	Group Art Unit:	2128
		§		
Filed:	September 18, 2006	§	Examiner:	Patel, Shambhavi K.
		§		
For:	Method for Software Emulation of a Computer Hard Disk	§	Atty Docket:	200800984-6 HPQB:0195
		§		

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF TRANSMISSION OR MAILING
37 C.F.R. 1.8

I hereby certify that this correspondence is being transmitted by facsimile to the United States Patent and Trademark Office in accordance with 37 C.F.R. § 1.6(d), or is being transmitted via the Office electronic filing system in accordance with 37 C.F.R. § 1.6(a)(4), or is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date below:

March 22, 2011	/Christopher R. Rogers/
Date	Christopher R. Rogers, Reg. No. 59,664

APPEAL BRIEF PURSUANT TO 37 C.F.R. §§ 41.31 AND 41.37

This Appeal Brief is being filed in response to the Final Office Action mailed on November 26, 2010, and in furtherance of a Notice of Appeal filed January 26, 2011.

1. REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 11445 Compaq Center Dr. W, Houston, TX 77070, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

2. RELATED APPEALS AND INTERFERENCES

The Appellants are unaware of any other appeals or interferences related to this Appeal. The undersigned is Appellants' legal representative in this Appeal.

3. STATUS OF CLAIMS

Claims 1-46 are currently pending, are currently under rejection and, thus, are the subject of this appeal.

4. STATUS OF AMENDMENTS

There are no outstanding amendments to be considered by the Board.

5. SUMMARY OF CLAIMED SUBJECT MATTER

The Application contains independent claim 1 and dependent claims 2-46, all of which are the subject of this appeal. As an example, independent claim 1 relates generally to a method which "totally emulates the software of hard disks at the level of the data blocks, also called sectors, or at the level of the file system, therefore permitting the use of file systems accepted by the operating system in emulated hard disks of any type." *See* Application, p. 3, ll. 4-7; *see also*, p. 1, ll. 7-14. In this Appeal Brief, the line number citations are provided to the English translation of the PCT Application as filed on September 18, 2006. The subject matter of independent claim 1 is summarized below. The subject matter of dependent claims 12, 16, 19, 20, and 41 is also summarized.

With regard to independent claim 1, discussions of the recited features can be found at least in the below-cited locations of the specification. By way of example, claim 1 recites a method that is performed by a suitably programmed processor. *See, e.g.*, p. 1, l. 19 – p. 2, l. 2. The method provides for the software emulation of hard disks of a data processing platform at the level of an operating system with parameterizable management of requests for writing and reading data. *See id.* at p. 1, ll. 15-18; p. 4, ll. 19-26; p. 4, l. 28 – p. 5, l. 2; p. 5, ll. 14-18; p. 7, ll. 15-23; p. 8, ll. 12-13 and 19-23 (discussing "order of priority"); p. 9, ll. 19-23; p. 9, l. 26 – p. 10, l. 11; p. 14, ll. 16-19. The method includes creating a representation of a real hard disk, wherein the sequence and location for loading and execution of components of the operating system of the data processing platform may be modified. *See id.* at p. 3, ll. 9-11; p. 13, ll. 1-7 and 12-18. The method also includes loading on said data processing platform one or more peripheral drivers, wherein at least one of the peripheral drivers communicates with a data storage support containing the data of the representation of the real hard disk. *See id.* at p. 3, ll. 11-14 and 16-19; p. 4, ll. 16-19 and 26-28; p. 7, ll. 3-12; p. 11, l. 27 – p. 12, l. 3. Further, the method includes simulating behavior of the real hard disk for the operating system, wherein the method transforms programming contained on the real hard disk into an emulated hard disk capable of controlling read and write operations on a client station and among two or more client stations. *See id.* at p. 3, ll. 13-15; p. 4, ll. 1-7 and 17-19; p. 4, ll. 26-28; p. 6, ll. 3-6; p. 11, ll. 8-26.

Dependent claim 12 recites a method as claimed in claim 1, wherein if the data support containing the data of the emulated hard disk is a support that does not provide for writing in real time, or does not accept writing of data directly in the support containing the data of the emulated hard disk, the data writing requests issued by the operating system to the emulated hard disk are processed in such a way that the written data are stored in a storage space different from the data support containing the data of the emulated hard disk. *See* Application, p. 2, ll. 10-14; p. 3, ll. 16-24; p. 7, l. 13 – p. 8, l.

17. The emulated hard disk may include one or more emulated hard disks. *See id.* at p. 2, ll. 10-12.

Dependent claim 16 recites a method as claimed in claim 1, wherein the data writing requests issued by the operating system to the emulated hard disk are redirected to a single storage space. *See* Application, p. 8, ll. 4-8. Further, the storage space in which the written data are redirected may be changed during an operating session of the operating system of a client station. *See id.* at p. 8, ll. 8-10.

Dependent claim 19 recites a method as claimed in claim 1, wherein the data reading requests issued by the operating system are performed in different storage spaces during an operating session of the operating system of a client station. *See* Application, p. 8, ll. 18-19.

Dependent claim 20 recites a method as claimed in claim 19, wherein the data reading requests issued by the operating system to an emulated hard disk carried out in different storage spaces follow an order of priority. *See* Application, p. 8, ll. 19-26.

Dependent claim 41 recites a method as claimed in claim 21, wherein if the data storage support containing the data of the emulated hard disk is a support that does not provide writing in real time, or does not accept write operations directly in the support containing the data of the emulated hard disk, the server program providing the emulation of the hard disk at the client stations processes the data write requests issued by the operating system to the emulated hard disk in such a way that the written data are stored in a storage space different from the data storage support containing the data of the emulated hard disk. *See* Application, p. 2, ll. 10-14; p. 3, ll. 16-24; p. 7, l. 13 – p. 8, l. 17.

6. SOLE GROUND OF REJECTION TO BE REVIEWED ON APPEAL

A. Sole Ground of Rejection for Review on Appeal

The Appellants respectfully urge the Board to review and reverse the Examiner's first ground of rejection in which the Examiner rejected claims 1-46 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Application Publication No. 2003/0200290 by Zimmerman, et al. (hereinafter "Zimmerman") in view of U.S. Patent No. 6,857,069 to Rissmeyer, et al. (hereinafter "Rissmeyer").

7. ARGUMENT

As discussed in detail below, the Examiner has improperly rejected the pending claims. Further, the Examiner has misapplied long-standing and binding legal precedents and principles in rejecting the claims under 35 U.S.C. § 103(a). Accordingly, the Appellants respectfully request full and favorable consideration by the Board, as the Appellants assert that claims 1-46 are currently in condition for allowance.

A. Sole Ground of Rejection

With respect to the rejection of claims 1-46 under 35 U.S.C. § 103(a) as being unpatentable over Zimmerman in view of Rissmeyer, the Examiner focused on claim 1, specifically stating:

> **Regarding claim 1:**
> **Zimmerman discloses** a method, performed by a suitably programmed computer, for software emulation of hard disks of a data processing platform at the level of the operating system with parameterizable management of requests for writing and reading data, the method comprising:
>
> a. creating a representation of a real hard disk wherein the location for loading and execution of components of the operating system of the data processing platform may be modified **([0019]: drives hold O/S; [0058]: drives may comprise any nonvolatile storage devices)** . . .

> **Zimmerman does not explicitly disclose** modifying the sequence for loading and executing of components of the operating system of the data processing platform may be modified. **Rissmeyer teaches** modifying the sequence for loading and executing of components of the operating system of the data processing platform may be modified **(column 1 lines 46-59: loading a network driver before loading a disk driver in an OS booting over a network)**. At the time of the invention, it would have been obvious to one of ordinary skill in the art to combine the teachings of Zimmerman and Rissmeyer because this allows the traditional controlling of devices prior to booting to occur without customized parts **(Rissmeyer: column 1 lines 11-44)**.

Final Office Action, pp. 4-5 (emphasis in original). The Appellants respectfully traverse this rejection.

The burden of establishing a *prima facie* case of obviousness falls on the Examiner. *Ex parte Wolters and Kuypers*, 214 U.S.P.Q. 735 (B.P.A.I. 1979). To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka,* 180 U.S.P.Q. 580 (C.C.P.A. 1974). Although a showing of obviousness under 35 U.S.C. § 103 does not require an express teaching, suggestion or motivation to combine prior art references, such a showing has been described by the Supreme Court as providing a "helpful insight" into the obviousness inquiry. *KSR Int'l. Co. v. Teleflex, Inc.*, 550 U.S. 398, 401, 82 U.S.P.Q.2d 1385, 1389 (2007). Moreover, obviousness cannot be established by a mere showing that each claimed element is present in the prior art. *Id.* The Examiner must cite a compelling reason why a person having ordinary skill in the art would combine known elements in order to support a proper rejection under 35 U.S.C. § 103. *Id.*

Zimmerman and Rissmeyer, alone or in any type of combination, fail to disclose all of the elements of claims 1-46.

Independent Claim 1

As explained in the present specification:

> In order to permit the startup from an emulated hard disk, *the sequence of loading of the components of the operating system may require an adaptation* so that all components of the operating system on which the peripheral drivers depend permit access to the emulated hard disk according to the invention are loaded and usable at the instant when the operating system needs to access the emulated hard disk by using the peripheral drivers and no longer using the "firmware" functions (BIOS).

Specification, p. 13, ll. 12-18 (emphasis added); *see also id.* at p. 1, ll. 10-18; p. 3, ll. 3-11; p. 4, ll. 23-25 ("This total emulation, including the possibility of startup, may require adjustments of the orders of loading and execution of certain of the operating system components."); p. 11, l. 27 – p. 12, l. 8; p. 13, l. 19 – p. 14, l. 3 (providing examples of how the sequence may be modified); p. 14, ll. 4-6 ("It may therefore occur that the order of loading of the components on which the peripheral drivers permitting access to the emulated hard disk depend will have to be modified.").

Accordingly, independent claim 1 recites, *inter alia*, "creating a representation of a real hard disk, wherein the *sequence* . . . for loading and execution of components of the operating system of the data processing platform may be modified." (Emphasis added). In the Office Action, the Examiner apparently relied on Rissmeyer to teach "modifying the *sequence* for loading and executing of components of the operating system . . . may be modified." *See* Final Office Action, p, 5 (citing Rissmeyer, col. 1, ll. 46-59).

In contrast, Rissmeyer discloses a fixed order of initially loading a network driver and then loading a disk driver. *See* Rissmeyer, col. 1, ll. 46-59. Rissmeyer does not disclose that this predetermined loading of components may be modified. Thus, Rissmeyer does not disclose that the order or sequence of loading and execution of components of an operating system changes or may be modified, as recited in claim 1.

Further, Zimmerman does not remedy this deficiency of Rissmeyer, nor did the Examiner allege so. Indeed, the Examiner admitted that Zimmerman does not disclose "modifying the sequence for loading and executing of components of the operating

system" (or that the sequence "may be modified"). *See* Final Office Action, p. 5. For these reasons, claim 1 and its dependent claims are patentable over the cited combination.

In the Response to Arguments section of the Final Office Action, the Examiner found Appellants' foregoing argument unpersuasive with regard to Rissmeyer, and maintained that Rissmeyer teaches that the sequence for loading and executing of components of the operating system may be modified, as recited in claim 1. The Examiner stated:

> **i. Applicant submits,** on pages 14-15, that Rissmeyer discloses a fixed order of initially loading a network driver and then loading a disk driver, but does not disclose that this predetermined loading of components may be modified.
>
> **Examiner notes** that Applicants appear to be arguing that [the] predetermined order or sequence of loading and execution of components of an operating system changes or may be modified *after* the real hard disk representation is created. However, the claims only require that they be modified/changed *in order to create* the real hard disk representation. Accordingly, Rissmeyer teaches modifying the sequence for loading and executing of components of the operating system when booting from a bootable operating system (see **column** 1 lines **46-59**) by loading a network driver before loading a disk driver.

Final Office Action, p. 2 (emphasis in original). Apparently, the Examiner asserted that because Rissmeyer loads a network driver before a disk driver (to facilitate booting from a second networked computer), that Rissmeyer somehow provides that the sequence of loading components of the operating system may be modified. However, the order of loading a network driver disclosed in Rissmeyer is fixed and must remained fixed for the Rissmeyer technique to function. *See* Rissmeyer, Abstract; col. 1, ll. 46-59; col 2, ll. 16-56; Fig. 1. In other words, Rissmeyer discloses a fixed and predetermined booting/loading of components that may *not* be modified. *See id.* Further, even if the Rissmeyer booting from a second networked computer, instead of from a local hard drive, could be characterized as a modification (which the Appellants do not concede),

Rissmeyer plainly does not provide for any flexibility to modify the fixed predetermined order of loading components of the operating system. Thus, Rissmeyer is contrary to claim 1.

In addition, independent claim 1 recites "creating a representation of a real hard disk, wherein the . . . *location* for loading and execution of components of the operating system components . . . may be modified." (Emphasis added). The Examiner apparently relied on Zimmerman to teach that the "*location* for loading and execution of components of the operating system components . . . may be modified." *See* Final Office Action, pp. 4-5 (citing Zimmerman. paras. [0019] and [0058]). However, Zimmerman merely discloses that "the address of a server to boot from" is provided to a client 2 PC upon power-up of the client 2 PC (from hibernation). *See* Zimmerman, para. [0058]. Zimmerman does not disclose that the *location* for loading and execution of components of the operating system components may be modified.

The Examiner disagreed and stated that "Zimmerman discloses that the address of the server that is to be booted from maybe be modified," and that "[t]his [is] equivalent to modifying the source location of the components." *See* Final Office Action, p. 2. However, the Appellants respectfully contend that Examiner misreads Zimmerman. While Zimmerman discloses that "the address of the server to boot from" is reported, no indication is given that this location can be modified. *See* Zimmerman, para. [0058]. Zimmerman does not disclose that the *location* for loading and execution of components of the operating system components may be modified.

Furthermore, Rissmeyer does not remedy this deficiency of Zimmerman. Rissmeyer is merely directed to "operation of iSCSI devices, ensuring that the network interface is available for use when the iSCSI disk driver is loaded." *See* Rissmeyer, col. 1, ll. 46-59. Rissmeyer does not disclose that the *location* for loading and execution of components of the operating system components may be modified. For at least these

additional reasons, claim 1 and its dependent claims are patentable over the cited combination.

Finally, claim 1 recites that the software emulation has "parameterizable management of requests for writing and reading data." Although this element is in the preamble, the preamble of claim 1 recites specific elements referenced in the body of the claim (e.g., "an operating system" and "a data processing platform") and, further, "is necessary to give life, meaning, and vitality to the claim." *Pitney Bowes, Inc. v. Hewlett-Packard Co.*, 182 F.3d 1298, 1305, 51 USPQ2d 1161, 1165-66 (Fed. Cir. 1999); *see also* M.P.E.P. § 2111.02. Thus, the element should be construed as if in the balance of the claim.

The parameterizable management of requests for reading and writing data is discussed in several places in the specification. For example, the specification states that "[t]he possibility exists on the level of said service of managing the written data in parameterizable form and of being able, in particular, to provide in a storage space, specific to the client station/virtual hard disk pair, storage space that is not the common storage space of all clients." Application, p. 14, ll. 16-19. Further, the specification notes that "[t]he volatility or persistence of the storage space used for the storage of the written data is parameterizable when this would make sense." *Id.* at p. 8, ll. 12-13. In contrast, Zimmerman is directed to the simultaneous transfer of data from a network server to one or more client devices, with no mention of parameterization. *See* Zimmerman, para. [0001]. Rissmeyer is directed to booting via iSCSI and a network, with no disclosure of parameterization of reading and writing data. *See* Rissmeyer, Abstract; col. 1, ll. 6-9. Neither Zimmerman nor Rissmeyer discloses parameterizable management of requests for reading and writing data as recited in claim 1. Accordingly, independent claim 1 is patentable over Zimmerman and Rissmeyer for at least this additional reason.

Dependent Claims 12 and 41

In addition, while the dependent claims are patentable by virtue of their dependency on allowable base claim 1, the dependent claims are also patentable because of the subject matter they separately recite. For example, dependent claim 12 recites "wherein if the data support . . . is a support that does *not provide for writing in real time*, or does *not accept writing of data directly* . . . the data writing requests issued by the operating system to the emulated hard disk are processed in such a way that the written data are stored in a storage space *different* from the data support containing the data of the emulated hard disk." (Emphasis added). Similarly, dependent clam 41 recites "wherein if the data storage support . . . is a support that does *not provide writing in real time*, or does *not accept write operations directly* . . . the server program . . . processes the data write requests . . . in such a way that the written data are stored in a storage space *different* from the data storage support." (Emphasis added).

Thus, dependent claims 12 and 41 require that *if* the data support (containing the data of the emulated hard disk) does *not provide for writing in real time*, or does *not accept writing of data directly* (*e.g.*, the data support is an optical CD-ROM disk), *then* written data are stored in a storage space *different* from the data support. *See* Application, p. 7, l. 13 – p. 8, l. 12; *see also id.* at p. 3, ll. 16-19; p. 4, l. 24 – p. 5, l. 13.

In contrast, the Zimmerman client may write to the server 4, or may write cache locally (at the client) to avoid possible corruption of data at the server 4. *See* Zimmerman, para. [0069] ("That is, the storage driver caches locally all writes so that the writes are never committed to the virtual driver, in order that different clients do not simultaneously write to the same virtual image and corrupt it."); Fig. 1; *see also* Final Office Action, pp. 7-8 and 13 (citing Zimmerman, para. [0069]). However, Zimmerman does not write to the local cache (or to a storage space different from the server 4), *in response to the server 4 unable to accept writing in real time or directly. See* Zimmerman, para. [0069]; Figure 1.

Indeed, Zimmerman makes no mention of the server 4 being unable to accept writing in real time or directly, contrary to the instant claims. Zimmerman also makes no mention of any response if the server 4 is unable to accept writing in real time or directly, much less a response to alter the storage destination if the server 4 is unable to accept writing in real time, as claimed.

The Examiner disagreed and pointed to the above-mention Rissmeyer queue process (for writes) that reduces corruption. *See* Final Office Action, pp. 2-3 (citing Zimmerman paras. [0067] and [0069]). However, Rissmeyer does not disclose that this queue process is implemented because the server is unable to accept writing in real time or directly. *See* Zimmerman paras. [0067] and [0069. In fact, this Zimmerman queue process is a preliminary step to avoid corruption and is independent of the server. The Appellants respectfully assert that once the data leaves the Zimmerman local queue, the data is accepted in real time and directly at the server. In any case, Zimmerman does not disclose that its server is unable to accept writing of data in real time or directly.

In sum, contrary to claims 12 and 41, Zimmer does not disclose that if the data support does not provide for writing in real time, or does not accept writing of data directly, then written data are stored in a storage space different from the data support. Further, Rissmeyer does not remedy these deficiencies of Zimmerman, nor did the Examiner assert so. For this additional reason, dependent clams 12 and 41 are patentable over the cited combination.

Dependent Claims 16 and 19

Dependent claim 16 recites "wherein the *storage space* in which the written data are redirected may be *changed during an operating session* of the operating system of a client station." (Emphasis added). *See* Application, p. 8, ll. 4-8 ("The storage space to which the written data are redirected may be changed on the spur of the moment during an operating session of the operating system of the client station."). Similarly, dependent claim 19 recites "wherein the data reading requests issued by the operating system are

performed in *different storage spaces during an operating session*." (Emphasis added). *See id.* at p. 8, ll. 18-19.

In contrast, Zimmerman does not disclose changing to a different storage space for read or write requests *during an operating session*. While Zimmerman discloses a "multi-server network" (*i.e.*, "one or more additional servers may be coupled to the network and may communicate with the first server 4 and client PCs 2"), Zimmerman does not disclose that a target server or storage may be changed (*i.e.*, read or write requests are redirected) during the same operating session. *See* Zimmerman, para. [0021]; Fig. 1.

The Examiner disagreed and asserted that Rissmeyer discloses a "multi-server system" and concluded that it possible that "the location of the requested data may be on any of the servers, thereby disclosing that the location of the read request in different storage spaces." *See* Final Office Action, p. 3 (citing Zimmerman, para. [0019]). However, Zimmerman does not *expressly* disclose changing to a different storage space for read or write requests during an operating session, nor did the Examiner assert so. Further, such a disclosure is not *necessarily present* in Zimmerman, as would be required to support an assertion of inherency. *See In re Robertson,* 169 F.3d 743, 49 U.S.P.Q.2d 1949 (Fed. Cir. 1999). Indeed, the Zimmerman technique can function properly in relying on a single server 4. Moreover, Rissmeyer does not remedy these deficiencies of Zimmerman, nor did the Examiner assert so. For these additional reasons, dependent clams 16 and 19 are patentable over the cited combination.

Dependent Claim 20

Dependent claim 20 recites "wherein the data reading requests issued by the operating system to an emulated hard disk carried out in *different storage spaces* follow an *order of priority*." (Emphasis added). *See* Application, p. 8, l. 20 – p. 9, l. 18. The Examiner cited paragraph [0062] of Zimmerman as disclosing these features. *See* Final Office Action, p. 9. However, paragraph [0062] of Zimmerman does *not* disclose an

emulated hard disk carried out in *different* storage spaces, much less that data reading requests follow an *order of priority* in such an arrangement. Instead, Zimmerman discloses that a *single* network server 4 streams data packets to clients 2. *See* Zimmerman, paras [0026]-[0027] and [0062]; Fig. 1. The streamed data does not originate from different storage spaces.

The Examiner asserted that "the Applicant has provided no justification for this [above] conclusion." *See* Final Office Action, p. 3, last 3 lines. The Examiner contended that in Rissmeyer when "a first client is selected as a Read Requestor, only that client is allowed to make a read request" and "thus that client's read request is given priority over other client's read requests." *See id.* However, this assertion by the Examiner indicates nothing about reading requests carried out *in different storage spaces*.

Moreover, with regard to the Rissmeyer O/S components specifically mentioned in the cited paragraph [0062, the *single* server 4 streams the Zimmerman O/S master boot record (MBR) in response to clients 2 requesting an O/S MBR download. *See* Rissmeyer, paras. [0019] and [0062]; Fig. 1. Plainly, the O/S MBR download is not streamed from *different* storage spaces, but from the single network server 4. *See id.* at para. [0062]. Furthermore, Zimmerman does not follow an *order of priority* in such streaming, much less an order of priority as with the presently-recited arrangement of different source storage spaces.

Instead, Zimmerman discloses that an invitation period is presented for clients 2 to register for the O/S MBR download. *See id.* The streaming module 26 of the network server 4 designates (based on "round-robin fashion") a first client 2 "that will be allowed to make a read request of the streaming module 26 to transmit to all the registered clients." *See id.* Such an approach including the registration of clients and the designation of a first client cannot be reasonably characterized as the recited *order of priority*. *See id.*

In sum, Zimmerman does not disclose "wherein the data reading requests issued by the operating system to an emulated hard disk carried out in *different storage spaces* follow an *order of priority*," as recited in claim 20. (Emphasis added). Moreover, Rissmeyer does not remedy this deficiency of Zimmerman, nor did the Examiner assert so. For these additional reasons, dependent claim 20 is patentable over the cited combination.

For at least the reasons discussed above, the cited references, alone or in any sort of hypothetical combinations, fail to disclose all of the elements of independent claim 1. Accordingly, independent claim 1 is allowable over these references. Thus, for at least the same reasons as discussed above, claims 2-45 which depend from claim 1, are allowable. Further, as discussed, at least dependent claims 12, 16, 19, 20, and 41 are patentable for the additional reason of the subject matter they separately recite. Accordingly, the Appellants respectfully request that the Board reverse the rejection of claims 1-46 under 35 U.S.C. § 103(a).

B. Request for Reversal of the Rejection

In view of the reasons set forth above, the Appellants respectfully request the Board to reverse the rejections of claims 1-46 under 35 U.S.C. § 103(a).

Conclusion

The Appellants respectfully submit that all pending claims are in condition for allowance. However, if the Examiner or Board wishes to resolve any other issues by way

of a telephone conference, the Examiner or Board is kindly invited to contact the undersigned attorney at the telephone number indicated below.

Respectfully submitted,

Date: March 22, 2011

/Christopher R. Rogers/
Christopher R. Rogers
Reg. No. 59,664
International IP Law Group, P.C.
(832) 375-0200

CORRESPONDENCE ADDRESS:

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
Fort Collins, Colorado 80528

8. APPENDIX OF CLAIMS ON APPEAL

1. Method, performed by a suitably programmed processor, for software emulation of hard disks of a data processing platform at the level of an operating system with parameterizable management of requests for writing and reading data, the method comprising:

creating a representation of a real hard disk, wherein the sequence and location for loading and execution of components of the operating system of the data processing platform may be modified,

loading on said data processing platform one or more peripheral drivers, wherein at least one of the peripheral drivers communicates with a data storage support containing the data of the representation of the real hard disk, and

simulating behavior of the real hard disk for the operating system, wherein the method transforms programming contained on the real hard disk into an emulated hard disk capable of controlling read and write operations on a client station and among two or more client stations.

2. Method as claimed in claim 1, wherein the management of said data write requests that the operating system sends to the emulated hard disk is accomplished at a peripheral driver level and/or at a level of an optional hard disk server service on a data processing network, the written data being stored according to the parameterization of said peripheral drivers and/or said service server of the hard disk on the network, and wherein the parameterization accommodates the written data being stored in any one of:

- the support containing the emulated hard disk;
- the memory, random access or virtual, accessible to the operating system using the emulated hard disk;
- a volatile storage space accessible to the operating system using the emulated hard disk;

- a non-volatile storage space accessible to the operating system using the emulated hard disk;
- a volatile storage space accessible to the server service of emulated hard disks on a data processing network; or
- a non-volatile storage space accessible to the server service of emulated hard disks on a data processing network.

3. Method as claimed in claim 1, wherein the management of the data reading requests that the operating system issues to the emulated hard disk is accomplished at a peripheral driver level and/or at a level of an optional hard disk server service on a data processing network, the readings of previously written data being performed in the storage space of any one of:

- the support containing the emulated hard disk;
- random access or virtual memory accessible to the operating system using the emulated hard disk;
- a volatile storage space accessible to the operating system using the emulated hard disk;
- nonvolatile storage space accessible to the operating system using the emulated hard disk;
- a volatile storage space accessible to the server service of emulated hard disks on a data processing network; or
- a non-volatile storage space accessible to the server service of emulated hard disks on a data processing network.

4. Method as claimed in claim 1, wherein the emulation of the hard disk provided to the operating system of the client station is accomplished by the agency of a single, monolithic peripheral driver which communicates with the operating system in the manner of a hard disk and which communicates with the support containing the data of said emulated hard disk in a manner specific to this support.

5. Method as claimed in claim 1, wherein the data of the emulated hard disk are accessible to the client stations via a data processing network, and wherein the emulated hard disk comprises one or more emulated hard disks.

6. Method as claimed in claim 1, wherein when an emulated hard disk is started up, the method further comprises providing a low level micro-software module to access the data contained in the emulated hard disk, wherein the operating system is started up at the client station.

7. Method as claimed in claim 6, wherein the data processing network comprises a plurality of client stations, the client stations using a bootup PROM, the method further comprising using the bootup PROM to control communications via the data processing network.

8. Method as claimed in claim 7, wherein the low level micro-software module is loaded in the memory of the client station and executed using the bootup PROM.

9. Method as claimed in claim 6, wherein the low level micro-software module is loaded in memory of the client station and executed as a component of the BIOS of the client station, said low level micro-software module providing the same functions as access services on real hard disks provided by the BIOS.

10. Method as claimed in claim 6, wherein the low-level micro-software is loaded in memory of the client station from a third party data support supported as a startup peripheral by the client station.

11. Method as claimed in claim 5, wherein at least one peripheral driver loaded and executed by the operating system of the client station provides the functions

of access, via the data processing network, to the data contained in the emulated hard disk.

12. Method as claimed in claim 1, wherein if the data support containing the data of the emulated hard disk is a support that does not provide for writing in real time, or does not accept writing of data directly in the support containing the data of the emulated hard disk, the data writing requests issued by the operating system to the emulated hard disk are processed in such a way that the written data are stored in a storage space different from the data support containing the data of the emulated hard disk, wherein the emulated hard disk comprises one or more emulated hard disks.

13. Method as claimed in claim 12, wherein the data writing requests issued by the client station operating system to the emulated hard disk are processed in such a way that the written data are stored in a random access memory of the client station.

14. Method as claimed in claim 12, wherein the data writing requests issued by the client station operating system to the emulated hard disk are processed in such a way that the written data are stored in a virtual memory of the client station.

15. Method as claimed in claim 12, wherein the data writing requests issued by the client station operating system to the emulated hard disk are processed in such a way that the written data are stored in a data file accessible to the operating system of the client station.

16. Method as claimed in claim 1, wherein the data writing requests issued by the operating system to the emulated hard disk are redirected to a single storage space, and wherein the storage space in which the written data are redirected may be changed during an operating session of the operating system of a client station.

17. Method as claimed in claim 12, wherein the storage space used for storage of the written data may be volatile, or may be nonvolatile so as to permit the written data of an operating session of the operating system to persist from one client station to another.

18. Method as claimed in claim 16, wherein a volatile character of the redirections of written data is determined upon initialization of the operating session of the operating system of the client station.

19. Method as claimed in claim 1, wherein the data reading requests issued by the operating system are performed in different storage spaces during an operating session of the operating system of a client station.

20. Method as claimed in claim 19, wherein the data reading requests issued by the operating system to an emulated hard disk carried out in different storage spaces follow an order of priority.

21. Method as claimed in claim 5, wherein a server program is in charge at one of the client stations of the data processing network, on the one hand, of the communications via the data processing network with the client stations accessing the emulated hard disks, and on the other, of accessing the data support containing the data of the emulated hard disk.

22. Method as claimed in claim 21, wherein if the hard disk emulation is parameterized so that the data write requests received by the server program are intended for a specific emulated hard disk they are not redirected but stored directly in a support containing the data of the emulated hard disk itself, and only one client station can access said emulated hard disk at a given time.

23. Method as claimed in claim 21, wherein in order to permit several client stations to access an emulated hard disk simultaneously, the server program is capable of redirecting specifically the data write requests issued by a client station A to a given storage space, and of redirecting the data write requests issued by another client station B to another given storage space.

24. Method as claimed in claim 21, wherein in order to permit startup from and/or simultaneous access to the same emulated hard disk or 100% identical copies of the same emulated hard disk, components of the operating system loaded and executed by the client stations or server program are capable of modifying, during or before their use by the operating system, data contained in the emulated hard disk.

25. Method as claimed in claim 1, wherein emulation is performed for the operating system of the client stations at a level of a class of virtual peripherals of a file system type.

26. Method as claimed in claim 1, wherein emulation is performed for the operating system of the client stations at the level of the class of disk peripherals itself and not at a file system level.

27. Method as claimed in claim 1, wherein data contained in the emulated hard disk are copied by a software tool executed at a client station from the real hard disk.

28. Method as claimed in claim 27, wherein the software tool creates an image directory that contains the data of the emulated hard disk.

29. Method as claimed in claim 27, wherein the software tool creates an image file that contains the data of the emulated hard disk.

30. Method as claimed in claim 1, wherein in order to permit startup from an emulated hard disk, the sequence of loading of the components of the operating system is modified so that all components of the operating system on which the peripheral drivers permitting access to the emulated hard disk depend are loaded and usable at the moment when the operating system accesses the emulated hard disk by using the peripheral drivers.

31. Method as claimed in claim 21, wherein in order to accelerate the simultaneous access by several client stations to the same emulated hard disk whose data are contained in a data support accessible to a server station, the data are sent by the server station to the client stations using broadcast or multicast mechanisms.

32. Method as claimed in claim 31, wherein the data sent by broadcast or by multicast by the server station are stored by the client stations that accept them in a local cache situated in the memory of said client stations.

33. Method as claimed in claim 31, wherein a read request for data in the emulated hard disk issued by the operating system of a client station generates an data reading request sent to the server station only if said data are not already present in local cache.

34. Method as claimed in claim 33, wherein the data read in the local cache are removed after being read by the client station so as to free up space in said local cache.

35. Method as claimed in claim 31, wherein a decision to send data by "multicast/broadcast" is made at a server module level which provides functionalities for the hard disk emulation at the client stations.

36. Method as claimed in claim 31, wherein the client stations may modify their subscription to receiving the data sent via "broadcast/multicast" by the server station.

37. Method as claimed in claim 32, wherein the client stations may erase the data from the local cache after a certain parameterizable time.

38. Method as claimed in claim 5, wherein data contained in a server module making the hard disks available to client stations may use any suitable network protocol including one of NVD, iSCSI, and SMB/CFIS.

39. Method as claimed in claim 5, wherein a low level software program executed by the client stations permits access to the data contained in the emulated hard disks using any suitable network protocol including one of NVD, iSCSI, and SMB/CFIS.

40. Method as claimed in claim 11, wherein at least one peripheral driver executed by the client stations permit access to the data contained in the emulated hard disks any suitable network protocol including one of NVD, iSCSI, and SMB/CFIS.

41. Method as claimed in claim 21, wherein if the data storage support containing the data of the emulated hard disk is a support that does not provide writing in real time, or does not accept write operations directly in the support containing the data of the emulated hard disk, the server program providing the emulation of the hard disk at the client stations processes the data write requests issued by the operating system to the emulated hard disk in such a way that the written data are stored in a storage space different from the data storage support containing the data of the emulated hard disk.

42. Method as claimed in claim 21, wherein the data write requests issued by the client station operating system to the emulated hard disk are processed in such a way that the written data are stored in a random access memory of a server station.

43. Method as claimed in claim 21, wherein the data write requests issued by the client station operating system to the emulated hard disk are processed in such a way that the written data are stored in a virtual memory of a server station.

44. Method as claimed in claim 21, wherein the data write requests issued by the client station operating system to the emulated hard disk are processed in such a way that the written data are stored in a data tile accessible to a server software.

45. Method as claimed in claim 21, wherein the storage space used for storage of the written data may be volatile, or may be nonvolatile so as to permit the written data of an operating session of the operating system to persist from one client station to another.

46. Method as claimed in claim 16, wherein a volatile character of the redirections of the written data is determined upon initialization of the operating session of the operating system of the client station.

9. **EVIDENCE APPENDIX**

None.

10. RELATED PROCEEDINGS APPENDIX

None.